Getting Started with SuperMap Objects .NET 6R

SuperMap Software Co., Ltd.
Beijing· China
Legal Statement

1. The copyright of this document is covered by SuperMap Software Co., Ltd. in accordance with the Copyright Law of the People’s Republic of China and the Universal Copyright Convention. If, without the written permission of the company, any part of the document shall not in any way or any reason be used, copied, modified, transmitted, or bundled with other products to be used, sold, tort reserved.

2. “超图”, “SuperMap”, and SuperMap are the registered trademarks of SuperMap Software Co., Ltd., protected by the Copyright Law of the People’s Republic of China. If, without the written permission of the company, the trademarks shall not in any way or any reason be used, copied, modified, transmitted, or bundled with other products to be used, sold, tort reserved.

3. This document represents no responsibilities of any supplier or agent. Without statement, SuperMap Software Co. Ltd. has right to do modifications to this document.

4. The copyright of trademarks mentioned in this document belongs to the corresponding companies. Without the written permission of these companies, the trademarks shall not in any way or any reason be used, copied, modified, or transmitted.

5. The concerned software products and the updated products hereinafter in this document are developed and sold by SuperMap Software Co., Ltd.
Hereby declare

SuperMap Software Co., Ltd.:
Add: 7/F Tower B, Technology Fortune Center, No. 8 Xueqing Road,
      Haidian District, Beijing, 100192, P. R. China
Tel: +86-10-82736655-4170
Fax: +86-10-82734630
HomePage: www.supermap.com
Sales: request@supermap.com
Tech Support: globalsupport@supermap.com

SuperMap Software welcomes all advices and suggestions from you.
Preface

What’s the purpose of this book?
This book is intended to make users quickly understand and grasp SuperMap Objects .NET, the new .NET component product released by SuperMap Software Co., Ltd. All contents will be presented concisely and clearly to guide first learners getting started with SuperMap Objects .NET. Here, a series of basic GIS functionalities are involved. After reading this book, you should learn how to:

- Add SuperMap Objects .NET components to a project.
- Open SuperMap workspace files.
- Develop a program to browse a map, such as zooming in, zooming out, panning, etc.
- Identify features.
- Query features by attributes.

Who can use this book?
This book introduces a simple C# program, developed in Microsoft Visual Studio 2005, to implement the basic functionalities of GIS. This book fits the developers who have C# programming experience. It helps to acquire SuperMap Objects .NET components if you know how to develop using SuperMap Objects COM.

What’s the content of this book?
There are two chapters in this book. Chapter One introduces the required installation environment, data and interfaces used in Chapter Two is also mentioned herein. Chapter Two describes how to add SuperMap Objects .NET components to a project and how to develop a program to browse a map.

If you have any other problems or suggestions in using our products, please visit our
home page: www.supermap.com, or directly contact our Support Center for more help.
Content

1 Introduction........................................................................................................................................... 1
  1.1 Brief introduction.................................................................................................................................. 1
    1.1.1. Installation.................................................................................................................................. 2
    1.1.2. Classes (including Controls) and interfaces (properties, methods) .................................... 3
    1.1.3. Data............................................................................................................................................ 4
  1.2 Brief summary ..................................................................................................................................... 5

2 Getting Started.......................................................................................................................................... 7
  2.1 Step One: Create a new project ........................................................................................................ 7
  2.2 Step Two: Add the SuperMap Objects .NET control and components ....................................... 8
  2.3 Step Three: Open a map .................................................................................................................. 10
    2.3.1. Programming .......................................................................................................................... 11
    2.3.2. Add the FormDemo_FormClosing event .............................................................................. 13
    2.3.3. Running ................................................................................................................................... 15
  2.4 Step Four: Browse the map ............................................................................................................... 15
    2.4.1. Programming .......................................................................................................................... 16
    2.4.2. Running ................................................................................................................................... 17
  2.5 Step Five: Query attributes by identifying features ........................................................................ 18
    2.5.1. Programming .......................................................................................................................... 19
    2.5.2. Running ................................................................................................................................... 21
  2.6 Step Six: Query features by SQL ...................................................................................................... 21
    2.6.1. Programming .......................................................................................................................... 22
    2.6.2. Running ................................................................................................................................... 24
1 Introduction

1.1 Brief introduction

SuperMap Objects .NET is a fundamental developing platform of the SuperMap GIS Universal series. It is based on .NET technology. It can be applied to all kinds of application development, particular GIS applications on the .NET platform. Implementing with C++/CLI, the most powerful .NET CLI programming language, SuperMap Objects .NET retains the extreme runtime performance necessary and features the overall .NET excellent capacities including quickly and easily writing, building, testing and deploying applications. SuperMap Objects .NET is not an upgraded version of SuperMap Objects, because it is a component product directly constructed on the basis of UGC (Universal GIS Classes, developed by SuperMap) and it has reconstructed the architecture, and at the same time regulated its interfaces.

SuperMap Objects .NET fully supports the .NET platform as well as various developing languages including C#, VB.net, C++/CLI, etc. and also has some new features:

- Supporting various .NET developing languages
- High performance of .NET developing
- High efficiency of UGC core

Flexible and practical components

- Simple and easy-to-use interfaces
- Embedded SDX+

You can develop the simplest GIS system in a short time following this book, to
realize browsing the map, querying attributes by identifying features and querying features by constructing SQL expression.

### 1.1.1. Installation

You can develop applications using SuperMap Objects .NET in Visual Studio 2005 (or higher). The requirements of the operating system and the configuration are listed as follows:

Operating system requirements supported:

- Microsoft Windows 2000 (SP4 series)
- Microsoft Windows Server 2003 (SP1 series)
- Microsoft Windows XP (SP2 series)
- Microsoft Windows Vista
- Microsoft Windows Server 2008
- Microsoft Windows 7

Minimum hardware requirements:

- CPU: 800MHz processor
- RAM: 256 MB
- Free disk space: 10 GB

Recommended hardware configuration:

- CPU: 2.00Ghz processor
- RAM: 1 GB or higher
- Free disk space: 40 GB

For the 3D effect displaying, please refer to the following hardware requirements:
Minimum hardware requirements:

- CPU: 2.00Ghz single core processor
- RAM: 512 MB
- Free disk space: 10 GB
- Video card: 128 MB

Recommended hardware requirements:

- CPU: 3.00Ghz single core processor/2.00Ghz dual core processor
- RAM: 2 GB
- Free disk space: 80 GB
- Video card: 3D graphic accelerator with 256MB RAM

NOTE: Do not run the 3D module in the 16 byte color environment.

Other software requirements:

- Microsoft .NET Framework 2.0 (Included in install package)
- Microsoft Visual Studio 2005
- Visual C++ 2008 redistributable package (Included in the installation package)
- About the installation of SuperMap Objects .NET 6R, please refer to the Installation Guide.

1.1.2. Classes (including Controls) and interfaces (properties, methods)

Tab 1 Classes and methods used in Chapter Two

<table>
<thead>
<tr>
<th>Classes</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
</table>

### 1.1.3. Data

The GIS information distributed by SuperMap Objects .NET is the map information. At first you need to prepare a map to begin the following operations by programming. Usually, you can use our professional GIS desktop platform SuperMap Deskpro or any client-side applications developed by SuperMap Objects to create a map. About the detailed procedure of how to create a map please refer to the relative documents and SuperMap Deskpro Help. In this book, we use the world map as an example which is located in the installation directory: `\SuperMap`
Objects .NET 6\SampleData\, where, a map named “world” is stored in the workspace file –World.smw. The demonstration project in this book will mainly use this map to perform all kinds of operations and queries.

The program in this book uses SuperMap Objects .NET components to open the map, and perform operations on it. The data used is the world map (World.sdb), the engine type is SDBPlus, and the main layers include World (the word map), Capital (the capitals of countries). The default map to be opened is the world map. you can change the number and the names of layers to be opened by modifying the codes. World map: World.sdb and World.sdd.

<table>
<thead>
<tr>
<th>Tab 2 Data illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
</tr>
<tr>
<td>World.smw</td>
</tr>
</tbody>
</table>

1.2 Brief summary

If this is the first time to develop a light GIS application using SuperMap Objects .NET, just follow the steps described in Chapter II to begin constructing a basic map application. The MapControl and other objects such as Workspace will be used.
2

Getting Started

2.1 Step One: Create a new project

The following steps show you how to create a project in Visual Studio 2005 IDE, and develop an application using SuperMap Objects .NET.

First, create a folder of the project, e.g. C:\Myproject;

Then, locate the World.smw file of the sample data in the default folder of the software installed, e.g. C:\Program Files\SuperMap\SuperMap Objects .NET 6R\SampleData\World\

Start Visual Studio 2005 IDE and create a new project (File > New… > Project) in the folder (C:\Myproject), then set the name of the new project as GettingStarted, as Illustration 1 Create shows:
2.2 Step Two: Add the SuperMap Objects .NET control and components

When installing SuperMap Objects .NET, the system will automatically register components in the VS2005 environment. When opening Toolbox, you can see the result as shown in Illustration 2:
Besides, you can add SuperMap Objects .NET components to Toolbox yourself by following the steps below:

First, right-click Toolbox and click **Add Tab**, then input the name of the tab, for example, SuperMap.

Second, right-click the newly created tab and click **Choose Items** to display the dialog box as shown in Illustration 3 Load components of SuperMap Objects .NET.
10

2 Getting Started

Select MapControl and Workspace components of SuperMap Objects .NET, and click OK.

### 2.3 Step Three: Open a map

First, Set the name of the form window to FormDemo. Double-click a Workspace control in ToolBox to display it on the FormDemo, naming it Workspace1. Double-click a MapControl in ToolBox to display it on the FormDemo, naming it MapControl1.

Then, add a ToolStript to the window and add a button to the ToolStript. The table below is the property setting and description of the button. (Keeping default values to other properties):
Tab 2 The property setting and description of the button

<table>
<thead>
<tr>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>toolStripOpen</td>
<td>Open a workspace</td>
</tr>
</tbody>
</table>

Where, the image of the button added in the ToolStript is stored in content of project ..\GettingStarted\GettingStarted\References.

At last, add an OpenFileDialog control into the FormDemo window, naming it openFileDialog1. The result of added the control is shown below:

Illustration 2 Add control to FormDemo

2.3.1. Programming

Here we take SDBPlus workspace as an example.

Before programming, import SuperMap.Mapping and SuperMap.Data in SuperMap
Objects .NET and in the code editor, add the following code at the top:

```csharp
using SuperMap.Data;
using SuperMap.Mapping;
using SuperMap.UI;
```

Double-click `toolStripOpen` in FormDemo, and enter the code editor. Or you can refer to the steps in 2.3.2 to add the `toolStripOpen_Click` event in FormDemo.

Fill the following code in the `toolStripOpen_Click` event so that it looks like this:

```csharp
//*****Open a workspace******
private void toolStripOpen_Click(object sender, EventArgs e)
{
    //Create a file dialog for loading a Workspace file.
    openFileDialog1.Filter = "SuperMap Workspace File(*.smw)|*.smw";
    //Determine whether the result of opening is ok; if so, implement the following operations.
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        //Avoid continuously opening workspace and causing program error.
        mapControl1.Map.Close();
        workspace1.Close();
        mapControl1.Map.Refresh();
        //Define the name of the workspace to be opened.
        String fileName = openFileDialog1.FileName;
        //Define WorkspaceConnectionInfo.
        WorkspaceConnectionInfo connectionInfo = new WorkspaceConnectionInfo(fileName);
        //Open workspace
        workspace1.Open(connectionInfo);
        //Connect MapControl and Workspace.
        mapControl1.Map.Workspace = workspace1;
    }
```

```csharp
```
// Determine whether there are maps in the workspace
if (workspace1.Maps.Count == 0)
{
    MessageBox.Show("There is no map in the workspace!");
    return;
}

// Open the map by name.
mapControl1.Map.Open("World Map_day");

// Refresh the map window.
mapControl1.Map.Refresh();

Note: If the Workspace object has been bound with MapControl, when executing Workspace.Close() and Workspace.Open(), be sure to call the corresponding method of MapControl: mapControl1.Map.Close().

2.3.2. Add the FormDemo_FormClosing event

After running the program, the disk space occupied by the program should be disposed. In this example, you should first dispose the disk space occupied by MapControl and then that by Workspace. Follow the steps below to add code to the FormClosing event in FormDemo.

1. Select View>Designer, and select FormDemo.

2. In Properties Window, click Events, as shown below:
Illustration 5 Add the FormDemo_Closing event

3. Select **FormClosing**.

4. Input `FormDemo_FormClosing` in the textbox, and press Enter. Then an event method named `FormDemo_FormClosing` is created and displayed in the View Code window. Add the following code:

```csharp
// First, dispose the disk space of MapControl, then of Workspace
private void FormDemo_FormClosing(object sender, FormClosingEventArgs e)
{
    mapControl1.Dispose();
    workspace1.Close();
    workspace1.Dispose();
}
```
2.3.3. Running

The running result is as following:

Illustration 6 Open a map

2.4 Step Four: Browse the map

It is convenient to perform the basic operations using SuperMap Objects .NET components, such as zooming in on a map, panning a map, viewing on a map, etc. The below takes zoom in, zoom out, zoom free, pan, and view an entire map as examples to illustrate how to use the SuperMap Objects .NET component to develop
an application.

First, add five buttons to the toolStrip. Set the properties as follows. (the other properties are default values):

<table>
<thead>
<tr>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>toolStripPan</td>
<td>Pan the map.</td>
</tr>
<tr>
<td>toolStripZoomIn</td>
<td>Zoom in on the map.</td>
</tr>
<tr>
<td>toolStripZoomOut</td>
<td>Zoom out on the map.</td>
</tr>
<tr>
<td>toolStripZoomFree</td>
<td>Zoom freely on the map.</td>
</tr>
<tr>
<td>(toolStripViewEntire) View the entire map.</td>
<td></td>
</tr>
</tbody>
</table>

Then double-click each button to add the following code in the code page.

2.4.1. Programming

```csharp
//Zoom in on the map.
private void toolStripZoomIn_Click(object sender, EventArgs e)
{
    mapControl1.Action = SuperMap.UI.Action.ZoomIn;
}

//Zoom out on the map.
private void toolStripZoomOut_Click(object sender, EventArgs e)
{
    mapControl1.Action = SuperMap.UI.Action.ZoomOut;
}

//Zoom freely on the map.
private void toolStripZoomFree_Click(object sender, EventArgs e)
{
    mapControl1.Action = SuperMap.UI.Action.ZoomFree;
```
private void toolStripViewEntire_Click(object sender, EventArgs e)
{
    mapControl1.Map.ViewEntire();
}

private void toolStripPan_Click(object sender, EventArgs e)
{
    mapControl1.Action = SuperMap.UI.Action.Pan;
}

### 2.4.2. Running

The running result is as follows:
2.5 Step Five: Query attributes by identifying features

First, add two buttons to the toolStrip. Set the properties as follows. (the other properties are default values):

<table>
<thead>
<tr>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>toolStripSelect</td>
<td>Select the features.</td>
</tr>
<tr>
<td>toolStripQueryProperty</td>
<td>Query attributes by identifying features.</td>
</tr>
</tbody>
</table>

Then, add a DataGridView control to the FormDemo window and name it as dataGridView1 and keep other properties with default values.
At last, double-click each button to add the following code in the code page.

### 2.5.1. Programming

```csharp
//Select the features.
private void toolStripSelect_Click(object sender, EventArgs e)
{
    mapControl1.Action = SuperMap.UI.Action.Select2;
}

//******Query attributes by identifying features.******
private void toolStripQueryProperty_Click(object sender, EventArgs e)
{
    //Get the selection.
    Selection[] selection = mapControl1.Map.FindSelection(true);
    //Determine whether the selection is empty or not.
    if (selection == null || selection.Length == 0)
    {
        MessageBox.Show("Please select the features to be queried!");
        return;
    }
    //Convert the selection to recordset with ToRecordset method.
    Recordset recordset = selection[0].ToRecordset();
    this.dataGridView1.Columns.Clear();
    this.dataGridView1.Rows.Clear();
    for (int i = 0; i < recordset.FieldCount; i++)
    {
        //Define and get the name of field.
        String fieldName = recordset.GetFieldInfos()[i].Name;
        //Add the name of field to columns of dataGridView control.
        this.dataGridView1.Columns.Add(fieldName, fieldName);
    }
```
/Initialize rows of dataGridView control.
DataGridViewRow row = null;
//According to the count of the selected recordset, add the information of selected features to dataGridView.
while (!recordset.IsEOF)
{
    row = new DataGridViewRow();
    for (int i = 0; i < recordset.FieldCount; i++)
    {
        //Define and get the value of field.
        Object fieldValue = recordset.GetFieldValue(i);

        //Add the value of field to corresponding rows of dataGridView.
        DataGridViewTextBoxCell cell = new DataGridViewTextBoxCell();
        if (fieldValue != null)
        {
            cell.ValueType = fieldValue.GetType();
            cell.Value = fieldValue;
        }
        row.Cells.Add(cell);
    }
    this.dataGridView1.Rows.Add(row);
    recordset.MoveNext();
}
this.dataGridView1.Update();
recordset.Dispose();
2.5.2. Running

Illustration 8 Query attributes by identifying features

2.6 Step Six: Query features by SQL

First, add Label, TextBox and Button to toolStript of the FormDemo window. Set their properties as follows (the other properties are default values):

<table>
<thead>
<tr>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>toolStripLabel1</td>
<td>A label named Query filter.</td>
</tr>
<tr>
<td>toolStripTextBox1</td>
<td></td>
</tr>
</tbody>
</table>
2.6.1. Programming

```csharp
//******Query features by SQL *****
private void toolStripSQLQuery_Click(object sender, EventArgs e)
{
    //Determine whether the text of toolStripTextBox1 is empty.
    if (toolStripTextBox1.Text.Length == 0)
    {
        MessageBox.Show("There is no query filter.");
        return;
    }

    //Define the count of layers in the current map window.
    Int32 layerCount = mapControl1.Map.Layers.Count;

    //Determine whether there are opened layers.
    if (layerCount == 0)
    {
        MessageBox.Show("Please open a vector dataset firstly!");
        return;
    }

    //Define the query conditions.
    QueryParameter queryParameter = new QueryParameter();
    queryParameter.AttributeFilter = toolStripTextBox1.Text;
    queryParameter.CursorType = CursorType.Static;
    Boolean hasGeometry = false;

    //Traverse each layer to implement querying multi-layers.
    foreach (Layer layer in mapControl1.Map.Layers)
    {
        //Get a vector dataset and convert it to DatasetVector type.
    }
}
```
```csharp
    DatasetVector dataset = layer.Dataset as DatasetVector;
    if (dataset == null)
    {
        continue;
    }
    // Query the specified vector dataset with the given query condition and get the attribute data.
    Recordset recordset = dataset.Query(queryParameter);
    // Determine whether there is any query result.
    if (recordset.RecordCount > 0)
    {
        hasGeometry = true;
    }
    // Add the queried data to the selection, and highlight the data.
    Selection selection = layer.Selection;
    selection.FromRecordset(recordset);
    recordset.Dispose();
    }
    // If no query result, the message pops up.
    if (!hasGeometry)
    {
        MessageBox.Show("There's no record satisfying the query condition or the query condition may be not correct!");
    }
    // Dispose the occupied space.
    queryParameter.Dispose();

    // Refreshes the map window.
    mapControl1.Refresh();
    hasGeometry = false;
```
2.6.2. Running

Type the query filter: SmID <25, and then click the button to query. The highlighted part on the map is the query result.

Illustration 9 Query features by SQL